JavaScriot et le DOM

Yann Rotella

Université de Versailles Saint Quentin en Yvelines
Université Paris-Saclay

1 avril 2021

JAVASCRIPT

LE DOCUMENT OBJECT MODEL

JAVASCRIPT

- interprété
- Typé dynamiquement
- faiblement typé
- orienté objet
- fonctionnel

JAVASCRIPT

- interprété
- Typé dynamiquement
- faiblement typé
- orienté objet
- fonctionnel

```
console.log('Bienvenue dans ce cours');
```

SYNTAXE BASIQUE

- Nombres : `0, 2, 1.2, 1e10, 0x3C, NaN, Infinity`
- Chaînes de c. : `"chr", 'chr', `chr``
- Bool : `true, false`
- `undefined, null`
- Objets : `{ val: 2.3, name: "Eric"}`
- Tableaux : `[ 1, ["b" , "a"]]`

VARIABLES

`let`

- JS faiblement typé
- portée au niveau du bloc

VARIABLES

`let`

- JS faiblement typé
- portée au niveau du bloc

`var`

- portée au niveau de la fonction

VARIABLES

`let`

- JS faiblement typé
- portée au niveau du bloc

`var`

- portée au niveau de la fonction

`const`

- constante, identifiant non réaffecté
- modification si objet mutable

CONDITIONS

```
if (x == 1) {
        ...
} else if (x == 2) {
        ...
} else {
}
```

CONDITIONS

```
if (x == 1) {
        ...
} else if (x == 2) {
        ...
} else {
}
```

ou bien

```
switch (y) {
case 3:
        ...
        break;
default:
        ...
}
```

BOUCLES

```
for ( let i = 0 ; i < 5 ; i++ ) {...}

let obj = { v1: 1, v2: 2, v3: 3}
for ( let x in obj ) { .. }

let tab = [1,2,3,4]
for ( let x of tab ) { .. }

while (cond) { ... }

do { ...}
while (cond)
```

EXECPTIONS

```
try {
        ...
{ catch (e) {
        if (cond) {...}
        else { ... }
} finally {...}
```

Opérateurs

Faible et forte comapraison

```
2 == '2';
2 === '2';
```

Opérateurs

Faible et forte comapraison

```
2 == '2';
2 === '2';
```

Appartenance

```
let o = { a: 1, b: 2}
'a' in o;
1 in o
```

Opérateurs

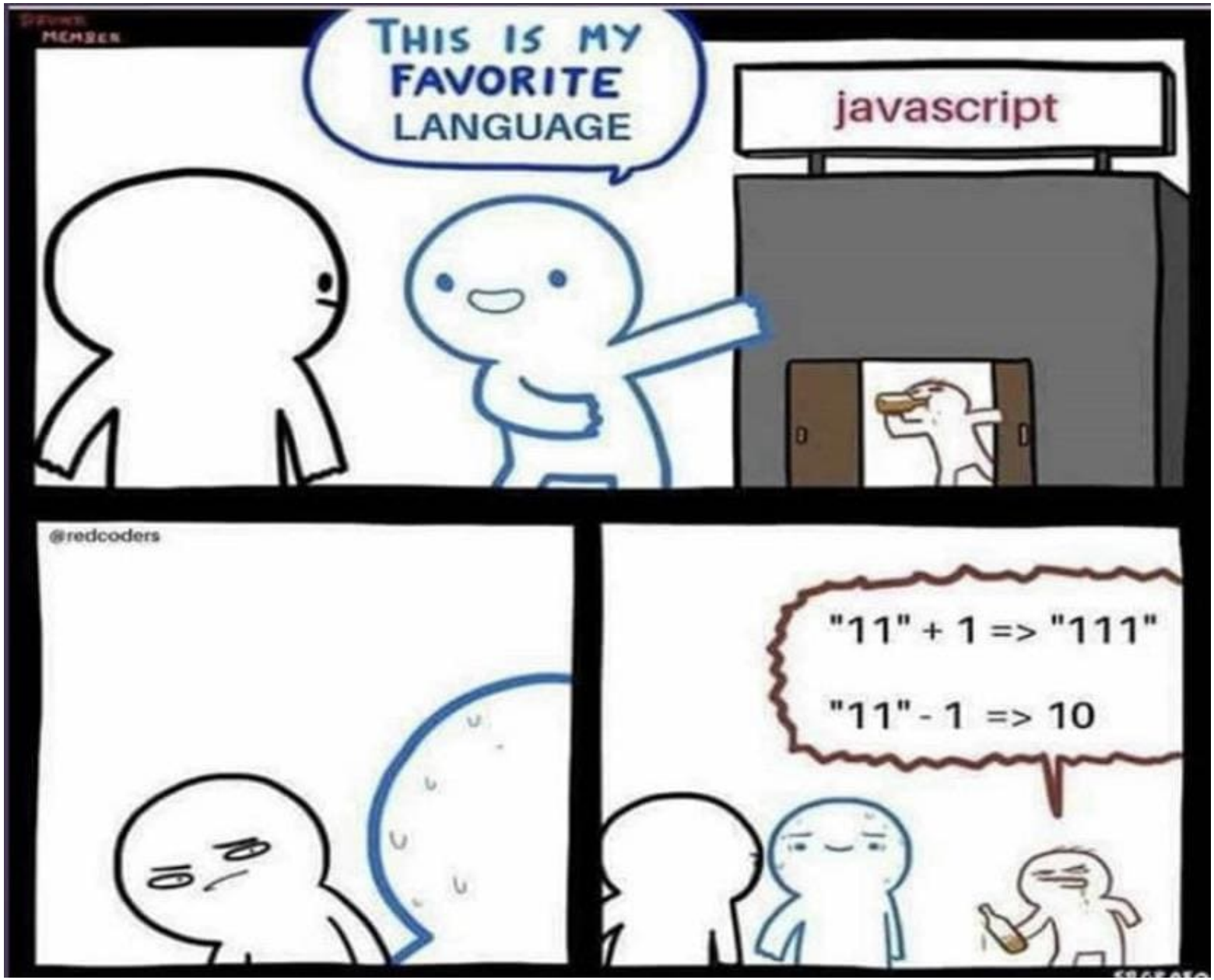
Faible et forte comapraison

```
2 == '2';
2 === '2';
```

Appartenance

```
let o = { a: 1, b: 2}
'a' in o;
1 in o
```

```
+, -, *, /, <, >, <=, >=, !, &&, ||, +
```


THIS IS MY FAVORITE LANGUAGE
javascript
@redcoders
"11" + 1 => "111"
"11" - 1 => 10

TABLEAUX

- allocation dynamique
- méthode `.length`
- `.join('string'), .indefOx(elem)`
- `map, filter, reduce, every, some, forEach, from, of, find, fill`

MÉTHODES USUELLES

- `push` : ajouter à la fin
- `pop` : supprimer de la fin
- `shift` : supprimer du début
- `unshift` : ajouter au début
- `concat`
- `includes` : cherche une valeur
- `reverse`, `slice`, `sort`

MÉTHODES UTILES

- `map` : applique une fonction
- `filter` : filtre avec une fonction
- `find` : idem (premier élément trouvé)
- `forEach` : applique une fonction

FONCTIONS

```
function f(x, y) {
        ...
}
```

FONCTIONS

```
function f(x, y) {
        ...
}
```

ou bien

```
let f = function (x, y){...}
let f = (x, y) => {...}
```

FONCTIONS

```
function f(x, y) {
        ...
}
```

ou bien

```
let f = function (x, y){...}
let f = (x, y) => {...}
```

On peut aussi ommettre des arguments avec `undefined` et mettre des valers par défaut :

FONCTIONS

```
function f(x, y) {
        ...
}
```

ou bien

```
let f = function (x, y){...}
let f = (x, y) => {...}
```

On peut aussi ommettre des arguments avec `undefined` et mettre des valers par défaut :

```
function f(x=0, y=2) {....}
```

FERMETURE DES FONCTIONS

```
function counter() {
        let c = 0;
        return function(step) {
        return c += step;
        }
}

let cnt = counter();
[cnt(1), cnt(2), cnt(1)]      // gives [1, 3, 4]
```

OBJETS

```
let myObj = {
        car: "Peugeot",
        color: "blue"
};

'car' in myObj;                 // true
myObj.car == "Peugeot";         // true
myObj['car'] == "Peugeot";      // true

let prop = 'car';
myObj[prop] == "Peugeot";       // true
delete myObj.color
```

Classes

Les classes `class` sont des fonctions qui produisent des objets. On a accès à `constructor`, et `new`.

Classes

Les classes `class` sont des fonctions qui produisent des objets. On a accès à `constructor`, et `new`. Une classe utile : `RegExp` (et `String`, `Boolean`, `Date`,...)

String methods

```
s = "Hello World"
```

- `s[4]`
- `length`
- `s.replace("Hello", "New")`
- `toUpperCase, toLowerCase`
- `indexOf`
- `slice(index_start, index_end)`
- `split`
- `includes, concat, repeat`

This

Différent selon l'utilisation.

- Dans une méthode, référence à l'objet
- tout seul, référence à l'objet global (window)
- dans une fonction, référence à l'objet global

LE DOCUMENT OBJECT MODEL

Interaction avec le HTML

Liaison objets et éléments HTML (arbre) : `HTMLelement`. Nous pouvons modifier les éléments, les attributs et le style dynamiquement.

- suppression et ajout
- réaction et création (événements)

RÉCUPÉRER LES ÉLÉMENTS

- `getElementById("ID")`
- `getElementsByClassName("classname")`
- `getElementsByTagName("p")`
- `querySelectorAll("p.classname")` ou `querySelector()`

MODIFIER LE HTML

- `.write()`
- `.innerHTML =`
- `.innerText =`
- `.attributeName =` (`.src`,...)
- `.classList` et `.classList.add` et `.remove`

Autres méthodes

- `.getAttribute("attributeName")`
- `.setAttribute("attributeName", "attribuiteValue")`
- `.hasAttribute(...)`
- `.createElement(nodeName)`
- `.appendChild()`
- `.append(...)`, `.prepend(...)`
- `.remove()`
- `.style.xxx` pour le css

Evénements

- `.addEventListener(..., fun);`
- `onload, onclick, onmouseover, onkeypress, ondrag,...`

Les propriétés des objets `event`

- `currentTarget` : l'élément
- `target` : l'élément déclencheur
- `type` : souris, clavier
- `which` : quelle touche
- et d'autres

Frameworks et librairies

- React.js
- Vue.js
- Angular.js
- Jquery

Choisissez celui qui vous plait !

- Utiliser `defer` dans le chargement du script dans le HTML
- `var $ = (s) => document.querySelector(s);`